Integrating Planning and Reaction
A Preliminary Report

John Bresina
Mark Drummond

AI Research Branch, Mail Stop 244-17
NASA Ames Research Center
Moffett Field, CA 94035

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA-91-20

January 1990

Integrating Planning and Reaction
A Preliminary Report

John Bresina and **Mark Drummond**
Sterling Federal Systems
AI Research Branch, Mail Stop: 244-17
NASA Ames Research Center
Moffett Field, CA 94035 U.S.A.

January 1990

Abstract

This paper is a preliminary report on the Entropy Reduction Engine architecture for integrating planning, scheduling, and control. The architecture is motivated through a NASA mission scenario and a brief list of design goals. The main body of the paper presents an overview of the Entropy Reduction Engine architecture by describing its major components, their interactions, and the way in which these interacting components satisfy the design goals.

This paper appears in *The Proceedings of the 1990 Stanford Spring Symposium*, in the session on Planning in Uncertain, Unpredictable or Changing Environments (March 27, 28, 29, 1990).

Integrating Planning and Reaction

A Preliminary Report

John Bresina and Mark Drummond*
Sterling Federal Systems
NASA Ames Research Center
Mail Stop: 244-17
Moffett Field, CA 94035

Abstract

This paper is a preliminary report on the Entropy Reduction Engine architecture for integrating planning, scheduling, and control. The architecture is motivated through a NASA mission scenario and a brief list of design goals. The main body of the paper presents an overview of the Entropy Reduction Engine architecture by describing its major components, their interactions, and the way in which these interacting components satisfy the design goals.

1 Motivation

NASA has plans to send a rover to Mars sometime this decade. Let's consider two extreme design scenarios for such a mission.

In the first scenario, let's assume that in advance of the rover's deployment, all relevant facts are known by the design team; for example, soil surface characteristics, surface topography, and location of all areas which could be hazardous to the rover. With all this foreknowledge, the designers can specify desired rover behavior for all situations the rover will encounter. The designers can produce a control system which enables the rover to achieve all scientific goals under Martian operating conditions.

Now consider a second scenario in which the design team has limited foreknowledge of the relevant facts needed to produce a rover control system. In this case, the rover must be capable of performing, on Mars, some of the activities that the designers could not complete due to lack of knowledge. For example, since the possible situations and goals will be unknown to the designers, the rover must be capable of determining, at runtime, a response appropriate to a novel situation-goal pair. This determination may involve synthesizing a complex behavior and evaluating it before acting.

*This work has been partially supported by the Air Force Office of Scientific Research, Artificial Intelligence Research Program.

These two scenarios vary only in the amount of foreknowledge possed by the rover designers. Most realistic mission scenarios will fall somewhere between these two – some parameters will be known in advance, and it will be necessary to determine some others at runtime. In any scenario there is a role for automated tools that reason about goals, that select actions relevant to those goals, that schedule selected actions, and that do temporal projection to determine possible consequences of behaviors. These tools can be useful as knowledge compilers in advance or as reactive systems at run time, or both to some degree, depending on the designers' foreknowledge and other mission constraints. Our research goal is to analyze, implement, and integrate such tools. The Entropy Reduction Engine (ERE) architecture is our developing body of theory in this endeavor.

2 Design Goals

The primary design goal for ERE has been to integrate planning (goal reasoning and action selection), scheduling (action sequencing and resource allocation), and control (monitoring of and adapting to a dynamic environment). This overall goal can be decomposed into the following design subgoals.

Manage goals with temporal extent. Standard planning goals of simple conjunctive achievement are not particularly useful in realistic situations. We want to be able to express behavioral constraints of maintenance and prevention over intervals of time.

Schedule actions in terms of metric time and metric resources. Most realistic applications for tools which manage time and actions involve a significant scheduling component. Planning and scheduling must be functionally integrated.

Synthesize plans. Scheduling a predetermined set of actions is not enough – many applications require that the set of actions be selected automatically.

Act without plans. It is not always possible to produce a plan for a problem in the time available. Unplanned action must be possible.

Manage disjunctive plans. The system must be able to represent and synthesize disjunctive plans. A

disjunctive plan is more robust; that is, it increases the likelihood of successful execution.

Reason about parallel actions. Parallelism is rife in realistic applications. Both possible and necessary parallelism must be handled in terms of representation and temporal projection capability.

Analyze plan execution as a control theory problem. A reaction plan can be viewed as a specification of how to react to a set of situation–goal pairs. Versions of this idea can be found in modern discrete event control theory (Ramadge and Wonham, 1989). These ideas from AI and control theory must be integrated and extended.

Encode problem solving strategies when available. Problem solving strategies for a domain or set of problems are often known by domain experts. We want to capture and exploit such expert knowledge so as to make search more efficient when possible.

Plan while things are changing. The world will often change while planning is going on. The plan formation process must be able to deal with changing situations.

Plan synthesis must have anytime, incremental characteritics. It should be possible to stop a plan synthesis algorithm at any time during its execution and expect useful results. One should also expect the "quality" of the results to improve continuously as a function of time. (Refer to Dean and Boddy, 1988 for more details.)

3 ERE Architecture Overview

This section gives a guided tour of our architecture and explains how it addresses each of the previous section's design goals. The ERE architecture includes the following components.

1. The *reactor* produces reactive behavior in the environment.
2. The *projector* explores possible futures and provides advice about appropriate behaviors to the reactor.
3. The *reductor* reasons about behavioral constraints and provides search control advice to the projector.

This architecture is organized around the *Principle of Independent Ability*, which is as follows: *each component must have the basic ability to perform its assigned task.* In no way does independent ability guarantee good performance; in fact, a component in isolation will typically exhibit poor performance and will improve only through interactions with other components.

For a concrete example consider the reactor and projector components. The reactor is able, in principle, to realize all the behaviors that are possible in a given domain. However, without any advice, the reactor is myopic – it does not know the future consequences of its behavior nor does it know whether its behavior will satisfy the given behavioral constraints. The performance level of the reactor is increased through interactions with the projector. The projector considers consequences of the various possible behaviors and advises the reactor on which particular behavior best satisfies the given behavioral constraints.

The reductor–projector interface is similar. Forward chronological search performed by the projector is inherently myopic; the projector does not have a "global picture" of the search space and as a result does not know which behaviors to project and which others to ignore. Of course projection can be done – it is just not very efficient. The projector aspires to efficiency by accepting search control guidance from the reductor. The reductor uses domain-specific planning expertise to recursively decompose the given problem into a conjunction of simpler (and more localized) subproblems. The conjunction represents a strategy for solving the overall problem and is used to provide global advice to the projector.

In both the projector–reactor and reductor–projector interactions, the input from one component simply serves to control an existing ability and does not serve to define that ability. This approach differs from that taken in classical "plan execution systems". A traditional plan executor (Wilkins, 1984) has nothing to do if it has no plan. In contrast, our reactor can always do something: the existence of a "plan" simply serves to increase the goal-achieving properties of the reactor. Similarly, the projector can consider possible futures without reference to some developing plan – search guidance from the reductor serves to control the projection when such advice is available, but such advice is not strictly necessary.

The principle of independent ability fits cleanly with the idea of an anytime algorithm. By decoupling the system into reduction, projection, and reaction, the ERE architecture can exploit each component's anytime characteristics. For example, the projector can give guidance to the reactor once it has found *a single* behavior satisfying the given constraints, and can incrementally augment this guidance with descriptions of other satisfactory behaviors as these are discovered in the projection.

The reductor has similar anytime characteristics. Initially, all behaviors which do not necessarily violate the overall behavioral constraint are allowed according to the reductor's first-cut problem solving strategy. Successive applications of reduction operators serve to refine the problem solving strategy providing search guidance that grows increasingly detailed and accurate over time, thus restricting the projector to ever fewer of the myriad possible behaviors.

The following three sections explain, in more detail, the functions of the ERE components and the nature of their decoupled anytime interaction.

3.1 The Reactor

The reactor accepts a specification of the environment's dynamics represented as a *plan net* (Drummond, 1985, 1986). A plan net defines the events that are possible in the environment in terms of each event's preconditions

and situation-dependent effects. Each event is represented by a single operator in the plan net. From the point of view of the reactor, a plan net can be characterized by a set of operators and the two functions given below, where S is the domain's set of possible situations, O is the set of plan net operators, and $\Pi(O)$ denotes the power set of O (note that this is a slight simplification of the full formalism explained in Drummond, 1989).

- $executancy : O \mapsto \{true, false\}$
- $enabled : S \mapsto \Pi(\Pi(O))$

The function `executancy` distinguishes between external events and agent-based actions. That is, `executancy(o)` indicates whether the reactor has control over the execution of the action denoted by operator `o` or whether `o` denotes an event whose occurrence is determined by the environment.

The function `enabled(s)` returns a set of operator sets in the plan net, where each of the operator sets returned can be performed in parallel in situation `s`. The reactor is only concerned with those operators that are enabled according to its current "world model". It needs to find a set of operators enabled in its world model for which it has executancy. The reactor interprets the plan net as a nondeterministic program, choosing and executing possible actions in an undefined order.

Control over the execution process is achieved by the use of Situated Control Rules, or SCRs (Drummond, 1989). An SCR is an if-then rule, where the antecedent refers to elements of the reactor's current world model and the current behavioral constraint, and where the consequent contains a set of possible operator sets to execute. Essentially, the consequent of an SCR for a situation s and behavioral constraint B contains those operator sets whose execution defines a prefix to a behavior which satisfies B. This means that the SCR's consequent is a subset of `enabled(s)`, since the operators that satisfy posted behavioral constraints will include some (but typically not all) of those operators that are enabled in s. The synthesis of these SCRs is discussed in more detail in Drummond (1989), and the next section provides a brief overview of the process.

The reactor always checks to see if any SCRs exist that are appropriate to the current situation and given behavioral constraints. If so, the SCRs' advice about what to do next is heeded. If there are no appropriate SCRs, unplanned execution is still possible. Without reference to the SCR input from the projector, the reactor simply selects and attempts to execute any enabled operator in the plan net. The results of such nondeterministic execution are (of course) unpredictable.

For the fully autonomous extreme of the rover example considered in section 1, the plan net given to the reactor would contain a specification of all actions the rover could perform, as well as all relevant external events which could affect the success of the rover's mission. For instance, an action for the rover could be `aim-laser-range-finder`, and an external event could be `rock-slips-from-gripper`. A background set of SCRs would be provided to give the rover essential reactions to situations demanding immediate response (*e.g.*, those needed for self-preservation). Other SCRs can be synthesized dynamically by the projector.

3.2 The Projector

The projection process considers the effects of events under the system's control and external events caused by the environment or other agents (*cf* Dean and McDermott, 1987). Projection is simply a search through the space of possible event sequences. A projection path represents a possible behavior. Considering all possible future behaviors is typically impossible.

The projector needs to view the plan net as a causal theory and so requires the following extra function which describes the effects of a set of operators o in a situation s. The function is defined $\forall o \subseteq O, s \in S$.

$$apply(o, s) = \begin{cases} s' \in S & \text{if } o \in enabled(s) \\ undefined & \text{otherwise} \end{cases}$$

Projection associates a duration with each set of operators applied and uses this to calculate a time stamp for each new situation. Currently, operator durations are integers and can be a function of the situation in which the operators are applied; situation time stamps are also integers.

Behavioral constraints are conjunctions and disjunctions of the following two forms.

- $(maintain\, \phi\, t_1\, t_2)$ is true of a projection path iff wff ϕ is true from time point t_1 through time point t_2 in the path.
- $(prevent\, \phi\, t_1\, t_2)$ is true of a projection path iff wff ϕ is false from time point t_1 through time point t_2 in the path.

A wff is a conjunction or disjunction of grounded predicates. Time points refer to situational time stamps and can be integers or variables; the domain of each variable is the integers. Arithmetic constraints on time point variables are allowed in the language. This language might appear quite simple but it allows us to express behavioral constraints that are more complicated than most planning systems can handle.

For example, the language allows the following:

```
(and   (maintain (memory 3 6) 1 5)
       (prevent (battery low) 2 7)
       (maintain (image taken) ?t ?t))
```

where `(memory 3 6)` indicates in our rover domain that the amount of memory available is between three and six megabytes, `(battery low)` indicates the battery's status, and `(image taken)` is true when a picture from the rover's camera has been taken. This constraint requires that the first predicate be true from time 1 through time 5 and that the second predicate be true from time 2 through time 7. The third conjunct in the constraint corresponds to a traditional goal of achievement, where the predicate must be true at an arbitrary but single point in time, here indicated by the variable `?t`.

Our approach calls for two phases of temporal projection. First, we find a single projection path that satisfies all given constraints. The search method used is based on likelihood (how probable is a candidate partial path; *cf* Hanks, 1990) and utility (how well does a candidate partial path satisfy the given constraints). The projection path is compiled into SCRs, giving the reactor a single correct behavior. The result of this first phase is somewhat like a triangle table (Nilsson, 1984) insofar as the reactor has information regarding what to do for any situation in a defined sequence. Our second phase of operation attempts to make this first solution more robust by strengthening probabilistically "weak" sections of the behavior. This two-phase approach gives the SCR synthesis anytime characteristics; details are explained by Drummond and Bresina (1990).

For a projection example let's look to our ongoing Mars rover scenario. There are limited resources on board, and given goals will often compete for these resources (e.g., the goals of obtaining a sample and of ensuring rover safety). Provided that an appropriate plan net and behavioral constraints are given to the on-board executive system, competing possible behaviors can be considered in terms of their likelihood and the degree to which each satisfies the given constraints. Projection will produce appropriate SCRs to be used by the reactor when the relevant situations arise.

The initial behavioral constraints will rarely provide enough control over the temporal projection search due to their scope: behavioral constraints are typically global, and temporal projection, while it eventually constructs a behavior with this global scope, does so incrementally through a series of single operator applications. Our problem of search control in this context is not new. All "goal-oriented" systems require a mechanism that can translate a computationally non-effective goal into a computationally effective means for controlling the search for a solution which satisfies the goal.

We expect the reductor to translate "global non-effective" behavioral constraints into ones that are "local" and "computationally-effective" to control temporal projection. The basic idea behind this translation process is the topic of the next section.

3.3 The Reductor

Standard problem reduction operates by applying nonterminal reduction rules to recursively decompose problems (situation–goal pairs) into conjunctions of "simpler" subproblems until "primitive" problems are recognized by terminal reduction rules which return their "obvious" solutions (Nilsson, 1971). A *complete* reduction trace is represented as an And tree whose root node represents the initial problem and whose leaves represent solved subproblems. The trace of a search through the reduction space is represented as an And/Or graph.

The ERE reductor is based on the REAPPR system (Bresina, 1988; Bresina, *et al.*, 1987) which extends this standard approach in a number of ways. REAPPR enables the encoding and effective utilization of domain specific and problem specific planning expertise. In order to fulfill its role in the ERE architecture, REAPPR is undergoing customizations and extensions.

In the ERE context, a problem is a pair consisting of a situation and a behavioral constraint. Nonterminal reductions can decompose a behavioral constraint based on its logical structure, its temporal extent, the logical structure of its formulae, or the semantics associated with the formulae's predicates.

For instance, in terms of the fully autonomous rover scenario, if a behavioral constraint requires that the distance to a nearby rock be precisely determined, then there might be two reductions giving more detailed behavioral constraints regarding how exactly this might be achieved. One reduction might specify that two visible light cameras should be used in conjunction with a calculation of binocular disparity; the other reduction might specify that the laser range finder should be used. The two alternative strategies have different costs and the reductions will indicate the situations under which each is appropriate.

The semantics of a nonterminal reduction is that satisfying the conjunction of behavioral constraints specified in the decomposition *implies* satisfaction of the original behavioral constraint. Furthermore, a nonterminal reduction represents the heuristic advice that satisfying the conjunctive subproblems is a *good strategy* for satisfying the original problem. By induction, given a partial reduction And tree, the set of leaf nodes represents a conjunction of subproblems whose satisfaction implies the satisfaction of the root node problem.

In accord with the standard approach, a terminal reduction applicable to a subproblem would return an action which is enabled in the subproblem's situation and satisfies the subproblem's behavioral constraints. Another use of terminal reductions is suggested by the following observation. Once a robust solution for a subproblem has been found by the projector and compiled into a set of SCRs, the projector no longer needs guidance from the reductor on solving subsequent occurrences of that particular subproblem. Hence, terminal reductions can be formed to recognize subproblems covered by existing SCRs, so the reductor will not waste time reasoning about them.

As the tree grows, the leaf subproblems become simpler and more localized; furthermore, they represent an increasingly accurate strategy for satisfying the initial problem. Hence, over time, the conjunctive set of leaf subproblems makes it increasingly easy to estimate the quality of a partial behavior in the projection and to estimate the likelihood that it can be extended to satisfy the overall constraints. The limit of this advice is a *complete* specification of all behaviors which satisfy the overall constraints. This limit is approached as more terminal reductions are applied.

4 Conclusion

We have implemented a temporal projection system based on the ideas outlined in this paper and have begun experiments in a domain loosely based around our autonomous Mars Rover scenario. This domain, *The Reactive TileWorld*, involves uncontrollable external events and the need to act before planning is complete. Behavioral constraints in the Reactive TileWorld are complex, typically involving the maintenance of conjunctions of predicates over intervals of time. We have implemented a subset of the goal language defined in this paper; in our language subset, if a variable is used to refer to the time points in a **maintain** or **prevent** statement, the same variable must be used for both the start point and end point. We have implemented the SCR compilation code defined in a previous paper (Drummond, 1989) and are currently developing a set of Reactive TileWorld benchmark experiments. The REAPPR system is being integrated with our temporal projection code.

How does our evolving architecture measure up in terms of our declared design goals? The architecture allows us to schedule actions in terms of metric time and metric resources by considering the situation-dependent effects of actions during projection. It also allows for synthesizing plans by selecting actions, for acting without plans, and for the management of disjunctive plans. The ERE architecture also supports reasoning about parallel actions in temporal projection. The reductor makes it possible to encode domain- and problem-specific strategies when such knowledge is available. All the components of our architecture have incremental, anytime properties. And what of our goal to plan while things are changing? We're working towards that by developing notions of situational *coverage* and overall system *robustness* in an effort to connect our work with modern discrete event control theory. Results will be reported in a forthcoming paper (Drummond and Bresina, 1990).

Acknowledgements

Thanks to other members of the ERE group for technical inspiration and for help with the implementation – Nancy Sliwa, Rich Levinson, and Andy Philips. Thanks also to Amy Lansky, Smadar Kedar, John Allen, and Ann Reid for helping with the paper and its production.

References

[1] Bresina, J. 1988. REAPPR – An Expert System Shell for Planning. Technical report LCSR-TR-119, LCSR, Rutgers University, February.

[2] Bresina, J., Marsella, S., and Schmidt, C. 1987 *Predicting Subproblem Interactions.* Rept. LCSR-TR-92, LCSR, Rutgers University, February.

[3] Bresina, J., Marsella, S., and Schmidt, C. 1986. REAPPR – Improving Planning Efficiency via Expertise and Reformulation. Rept. LCSR-TR-82, LCSR, Rutgers University, June.

[4] Dean, T., and Boddy, M. 1988. An Analysis of Time-Dependent Planning. In proc. of *AAAI-88.* pp. 49–54.

[5] Dean, T., and McDermott, D. 1987. Temporal Database Management. AI Journal, Vol. 32(1). pp. 1–55.

[6] Drummond, M. 1989. Situated Control Rules. *Proceedings of Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Canada.

[7] Drummond, M. 1986. A Representation of Action and Belief for Automatic Planning Systems. *Reasoning About Actions and Plans*, M. Georgeff and A. Lansky, Eds., Morgan Kauffman. pp. 189-211.

[8] Drummond, M. 1985. Refining and Extending the Procedural Net. *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Los Angeles, CA.

[9] Drummond, M., and Bresina, J. 1990. Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction. Submitted to *AAAI-90.*

[10] Hanks, S. 1990. Projecting Plans for Uncertain Worlds. Yale University, CS Department, YALE/CSD/RR#756.

[11] Ramadge, P.J.G., and Wonham, W.M. 1989. The Control of Discrete Event Systems. *Proceedings of the IEEE.* Vol. 77, No. 1 (January). pp. 81–98.

[12] Nilsson, N. 1971. Problem Solving Methods in Artificial Intelligence. McGraw Hill, N.Y.

[13] Nilsson, N. (ed). 1984. Shakey the Robot. Technical Note 323, Stanford Research Institute.

[14] Wilkins, D. 1984. Domain Independent Planning: Representation and Plan Generation. *Artificial Intelligence*, Vol. 22, pages 269–301.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Dates attached	

4. TITLE AND SUBTITLE

Titles/Authors - Attached

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Code FIA - Artificial Intelligence Research Branch
Information Sciences Division

8. PERFORMING ORGANIZATION REPORT NUMBER

Attached

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Nasa/Ames Research Center

Moffett Field, CA. 94035-1000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Available for Public Distribution

Peter Friedland 5/14/92 BRANCH CHIEF

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstracts ATTACHED

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT